Novia Raseborg-mobilapplikationen implementerad med Java

Robert Toivonen

Examensarbete för Tradenom (YH)-examen

Utbildningsprogrammet för INFORMATIONSBEHANDLING

Raseborg 2014

EXAMENSARBETE

Författare:	Robert Toivonen
Utbildningsprogram och ort:	Informationsbehandling Raseborg
Handledare:	Rolf Gammals

Titel: Novia Raseborg-mobilapplikationen implementerad med Java

---

Datum: 03.12.2014 Sidantal: 33 Bilagor: 4

Abstrakt

Examensarbetet redogör förverkligandet av en Androidapplikation med Java programmeringsspråket. Uppdragsgivarens syfte med Novia Raseborgapplikationen är att utveckla ett nytt sätt att förmedla information till studerande och skapa en ny kommunikationsmetod mellan studerande och studiehandledare. Meningen med detta är att undvika studieproblem hos studerande. Med detta projekt har uppdragsgivaren för avsikt att skapa en bättre Androidversion av applikationen Novia Raseborg. Uppdragets krav är att den nya versionen av applikationen skall ha samma funktioner som den gamla versionen. Applikationen skall också använda sig av samma externa information som den gamla versionen och skall därför kunna hantera denna information.

Eftersom applikationen inte skall ha några märkvärdiga visuella förändringar används de gamla bildfilerna. Inbyggd Androidfunktionalitet används där det är möjligt för att optimera applikationen. För att behålla utseendet oförändrat presenteras texterna i HTML-formatet. Matlistans utseende ändras från en enkel lista till en mera interaktiv och därmed mera användarvänlig lista.

Uppdraget genomfördes framgångsrikt och resulterade i en Androidversion av Novia Raseborgapplikationen utförd i Java. Den nya applikationen är märkbart snabbare än den föregående versionen som skapades med PhoneGap.

Språk: Svenska Nyckelord: Android, Applikation, Java, Mobilapplikation

---

BACHELOR'S THESIS

Author:	Robert Toivonen
Degree Programme and location:	Business Information Technology Raseborg
Supervisors:	Rolf Gammals

Title: Novia Raseborg mobile application implemented with Java / Novia Raseborg-mobilapplikationen implementerad med Java

---

Date: 3 December 2014 Number of pages: 33 Appendices: 4

Summary

The thesis describes the production of an Android application in the programming language Java. The commissioner´s request concerning this project is to get a better Android version of the application Novia Raseborg. Through this application a new method of informing the students at Novia will be established. It will also mean an improved communication between students and the study counsellor. The intended effect of this is to pre-empt study-related issues.

The assignment's requirement is that the new application ought to have the same functions as the old application. The application will need to use the same external information as the old version and therefore it has to be backwards compatible.

Since the application intends to remain visually similar to the old version, the image files are reused. The Native Android functionality is used where possible to optimize the application. In order to maintain the look of the old version the texts are presented in HTML. The Lunch list was changed from a simple list to a more user-friendly interactive list.

The assignment was successfully completed and resulted in an Android version of the Novia Raseborg application written in Java. The application is noticeably faster than the previous version that was constructed with PhoneGap. The functions of the new version are mostly the same as in the previous version. Thus, the expectations and demands have been met.

Language: Swedish Key words: Android, Application, Java, Mobile application

---

Innehållsförteckning

1 Inledning

Detta examensarbete redogör för utvecklingen av en Androidapplikation i programmeringsspråket Java. Projektets syfte var att skapa en bättre version av applikationen Novia Raseborg för Androidplattformen. Den förra versionen av applikationen var utförd i PhoneGap och visade sig något långsam. Genom att programmera en ny version i Java förväntade man sig att applikationen skulle fungera snabbare. Den nya versionen skulle inneha samma funktioner som sin föregångare. I detta projekt användes huvudsakligen den gamla designen. Innehållet som presenteras i den nya applikationen baserar sig på den föregående versionens informationssystem.

1.1 Bakgrund

Novia Raseborgapplikationen hade haft två versioner före detta projekt påbörjades.
Den första versionen var en mobilapplikation utvecklad i Flash. Denna första version var mycket enkel och hade några sidor med text om Yrkeshögskolan Novia och dess utbildningsprogram.

Arbetet på den andra versionen av Novia Raseborgapplikationen påbörjades år 2013. Syftet var att utveckla nya funktioner som gav tillgång till matlistor, studietips, schema, väderinformation, aktuella nyheter och e-post. Det var också meningen att versionen skulle vara tillgänglig till mobiltelefoner som var baserade på Android, iOS och Windows operativsystemen.

För att utveckla applikationen till alla valda plattformer valdes PhoneGap som skulle generera alla plattformernas versioner från samma HTML-, CSS- och JavaScriptfilerna. Det visade sig dock att PhoneGap inte fungerade så bra som förväntat och applikationen programmerades med hjälp av PhoneGap endast till Androidplattformen.

1.2 Syfte

Uppdragsgivarens syfte med applikationen var att skapa ett verktyg för att stöda studiehandledningen. Applikationen skulle som utgångspunkt skapa en kontakt mellan studerande och studiehandledaren. Ursprungliga funktionaliteten för applikationen kom

från vad studerande på Yrkeshögskolan Novia Raseborgs Campus ansåg viktigt. Uppdragsgivaren ansåg att med ett gemensamt digitalt verktyg skulle man kunna förebygga studieproblem som till exempel att studerande avbryter studierna (Hillo 2014, s. 81-82).

Syftet med detta projekt var att bygga en Androidversion av Novia Raseborgapplikationen i programmeringsspråket Java. Förväntningarna var att applikationen skulle vara snabbare och fungera likadant på de flesta mobiltelefoner med Androidoperativsystemet.

Applikationens utseende förblev det som den senaste versionen med uppdateringar där det var nödvändigt. Alla PNG-filer överfördes från förra versionen i den mån som de behövdes.

Funktionalitet skulle behållas liknande som i den förra versionen av applikationen. Det gamla systemet att spara information var ostrukturerat och därför skapades en databasstruktur för att spara information för applikationen.

2 Verktyg

För att skapa en Androidapplikation i Java behövs vissa verktyg. En integrerad utvecklingsmiljö hjälper att hålla reda på projektets alla filer och presentera dem i en användbar hierarki. Integrerade utvecklingsmiljöer hjälper också att provköra och omarbeta programmeringsprojekt tryggt.

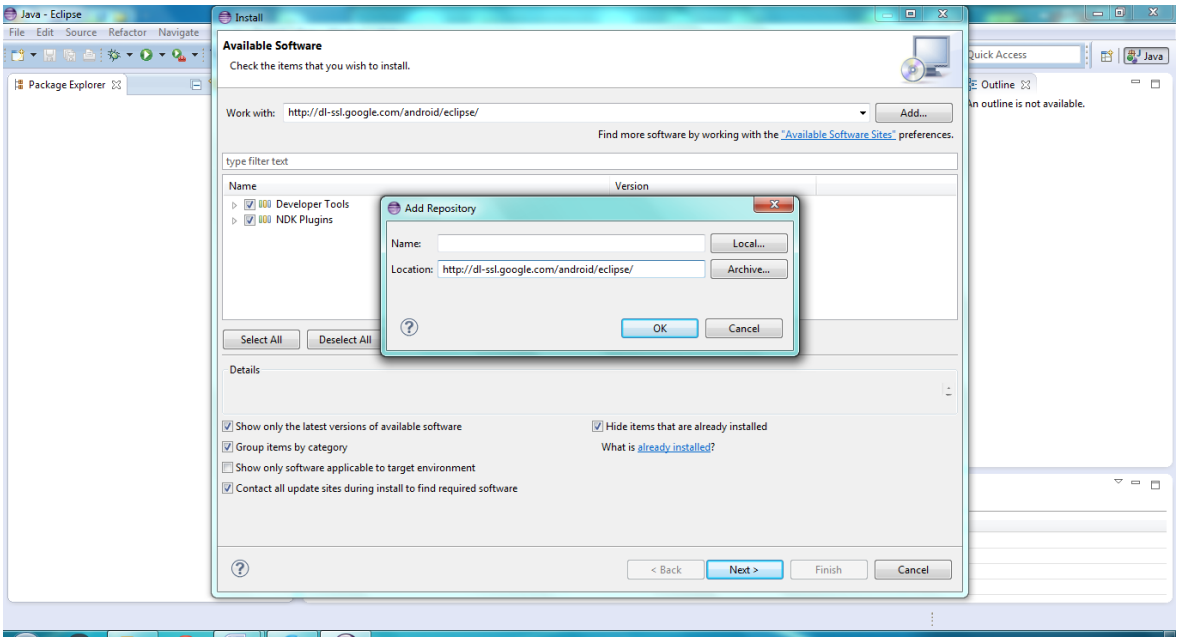
2.1 Eclipse

I detta projekt användes huvudsakligen Eclipse för att bearbeta filer och testa applikationen under projektets gång. Eclipse är en integrerad utvecklingsmiljö som är populär bland Java programmerare. Eclipse är tillgänglig från producentens hemsida, www.eclipse.org, under ”downloads” fliken. Att använda och skapa program med Eclipse kostar inget. Eclipse är enkelt att ta i bruk. Programmet kommer i en ZIP-fil, efter att programmet packats upp är det färdigt att användas. Då programmet startas upp första gången ber Eclipse användaren

att specificera en ”Workbench”. Detta är en mapp där alla projekt och deras filer sparas på hårdskivan.

För att programmera för Android med Eclipse rekommenderas insticksprogrammet ADT. Programmet förenklar skapandet av Androidprojekt. ADT insticksprogrammet håller reda på alla resurser som användaren producerar och som finns i Androidbiblioteket.

ADT insticksprogrammet kan laddas ner gratis via Eclipse. Programmet installeras under menyn Help > Install New Software, klicka på ”Add” detta öppnar ett nytt fönster med titeln ”Add Repository” (se Figur 1). Skriv in http://dl-ssl.google.com/android/eclipse/ i ”Location” fältet och klicka ”OK” välj alla program under Developer Tools och NDK Plugin och gå igenom resten av installationsprocessen. Efter att ”Finish” klickats installerar Eclipse ADT insticksprogrammet.



Figur 1. Installationsrutan för att söka program.

I detta skede är Eclipse inte ännu färdig för utvecklingen av Androidapplikationer. För att utveckla bör Android SDK Manager installeras som ger tillgång till Androidbiblioteken och resurserna för de olika versionerna av Android.

2.2 Android SDK Tools

För att ladda ner resurser för de olika versionerna av Android och emulera dem krävs Andriod SDK Tools. På Androids webbsidor för utvecklare kan man ladda ner dessa verktyg, de finns under ”USE AN EXISTING IDE” fliken på sidan developer.android.com/sdk.

När Android SDK Tools har installerats har användaren tillgång till Android SDK Manager och Android Virtual Device Manager. Android SDK Manager låter användare ladda ner resurser till de olika Androidversionerna. Android Virtual Device Manager kan användas för att skapa virtuella representationer av mobiltelefoner och emulera dessa. För att programmera en applikation till en eller flere av Androidversionerna måste man först ladda ner resurser för dessa med Android SDK Manager. Resurserna krävs också för att emulera virtuella mobiltelefonerna med respektive Androidoperativsystem.

2.3 ADT Bundle

Android erbjuder ett nerladdbart paket som innehåller både Eclipse med ADT plugin och Android SDK Tools. Detta paket kan hittas på Androids webbsidor för utvecklare, developer.android.com/sdk. ADT Bundel är behändigaste sättet att ställa upp utvecklingsmiljön för Android applikationer.

3 Design

I detta avsnitt beskrivs applikationens XML-kod och Javakoden som påverkar applikationens design för att redogöra hur applikationens utseende förverkligades. Utseendet på applikationen ändrades inte mycket under detta projekt och på grund av detta användes många grafiska element från förra projektet. För att kunna utnyttja dessa grafiska element i Java måste en elementstruktur programmeras.

3.1 XML-layout

För att manipulera grafiska element i Java måste de grafiska elementen först ställas upp i en XML-struktur. I XML-strukturen kan man definiera och deklarera element som är inbyggda i Android. Elementen faller under två huvudkategorier, de så kallade view och layoutelementen. XML-filerna placeras i mapphierarkin under res/layout/. Viewelementen används för att manipulera vad och hur information visas medan layoutelementen används för att skapa en elementstruktur.

LinearLayout är ett element som håller andra element som till exempel knappar inom sig och låser dem i den ordning de är deklarerade. Hur de visas beror på hur LinearLayout elementet har definierats. I kodsnutten nedan (se Kod 1) kan man se några exempel på parametrar som kan definieras för LinearLayout element.

Kod 1. LinearLayout exempel.

```
<LinearLayout
      android:layout_width='fill_parent'
      android:layout_height='wrap_content'
      android:orientation='vertical'
      android:id='@+id/exempelID'
>
</LinearLayout>
```

Kod raderna ”android:layout_width='fill_parent'” och ”android:layout_height='wrap_content'” definierar hur LinearLayout elementets innehåll skall placeras inom elementet. Då ”fill_parent” parametern deklareras för LinearLayout elementet använder den sig av sin maximala utsträckning i valda dimensionen. Då ”wrap_content” parametern deklareras använder LinearLayoutelementet den minsta mängden utrymme som är möjligt i valda riktningen. Linjen ”android:orientation='vertical'” fyller elementen inom LinearLayoutelementen i vertikala riktningen alternativt kan ”android:orientation='horizontal'” deklareras för att fylla i elementerna i horisontella riktningen. Man kan också definiera en id för XML-elementerna för användning i Java koden. Exempel på detta kan man se i Kod 1 med raden ”android:id='@+id/exempelID'”. **Error! Reference source not found.**Figur 2 illustrerar hur LinearLayoutelementet ordnar sitt innehåll i praktiken.

Figur 2. Praktiskt exempel på LinearLayoutelementet.

FrameLayout elementet liknar LinearLayout elementet. Detta element fungerar ofta som roten i XML-strukturen. Till skillnad från LinearLayoutelmentet, som ställer innehållet i ordning och låter elementen inte överskrida varandra, ställer FrameLayoutelementet sitt innehåll endast i nivåer som fritt kan röra sig på varandra. FrameLayout är därför använd när det behövs flere skikt av element så som i navigationsmenyn. FrameLayout elementet har liknande parametrar som LinearLayout och deklareras på liknande sätt (se Kod 2).

Kod 2. FrameLayoutelement struktur exempel.

```
<FrameLayout
      android:layout_width='fill_parent'
      android:layout_height='wrap_content'
      android:orientation='vertical'
      android:id='@+id/exempelID'
>
</FrameLayout>
```

Figur 3 demonstrerar hur Kod 2 ser ut i praktiken. I figuren kan man see hur alla bildelementen hamnar på varandra. Det första elementet som placeras i ett FrameLayoutelement blir alltid lägst i högen och resten staplas på i ordningsföljd.

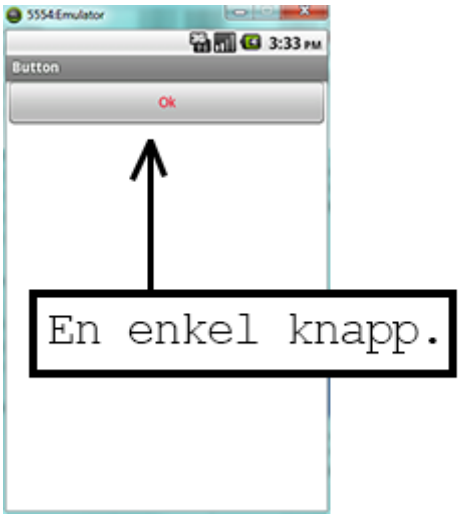
Figur 3. Praktisk illustration av FrameLayoutelementet.

Buttonelementen är enkla knappar vars grafik skapas av Androidplattformen. De har en begränsad mängd valbara designegenskaper. Dessa egenskaper är till exempel text, text färg och knappens dimensioner vilket framgår i Kod 3.

Kod 3. Buttoelement struktur exempel.

```
<Button
      android:text='Ok'
      android:textColor='#ff0026'
      android:layout_width='fill_parent'
      android:layout_height='wrap_content'
      android:id='@+id/exempelID'
/>
```

Figur 4 demonstrerar hur en vanlig knapp med inställningarna från Kod 3 ser ut i praktiken. I detta fallet är knappens text röd för att demonstrera koden men om ingen färg deklareras är texten svart. I praktiken används dessa knappar oftast för att antingen bekräfta något eller för att få ett ja eller nej svar på en fråga.



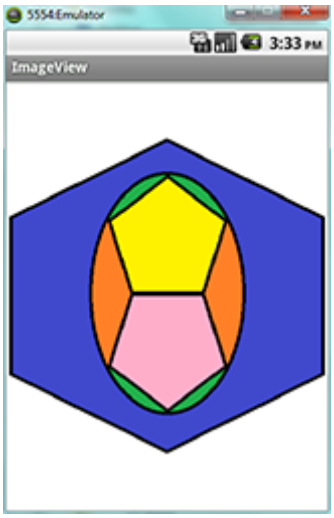
Figur 4. Demonstration av Buttonelementet.

ImageView elementet håller en bild. Bilderna refereras från Androids inbyggda mapphierarkin från mappen ”drawable”. Exempel på detta kan ses i Kod 4 på linjen `”android:src='@drawable/bildFilnamn'”`. Bildelementet kan skalas på olika sätt. Den kan till exempel skalas att passa in i elementet den befinner sig i.

Kod 4. ImageViewelement struktur exempel.

```
<ImageView
      android:layout_width='fill_parent'
      android:layout_height='fill_parent
      android:id='@+id/exempelID'
      android:src='@drawable/bildFilnamn'
/>
```

Figur 5 demonstrerar hur ett ImageViewelement förekommer med inställningarna från Kod 4. Dessa bildelement används för många olika skäl i Androidapplikationer. I Novia Raseborg används bildelement endast för navigationsmenyn. ImageViewelement kan användas på många sätt till exempel för bildgallerier.



Figur 5. Praktisk demonstration av ImageViewelementet.

ImageButton är ett element som är en kombination av Button och ImageView elementen. ImageButtonelementet används för att hålla en bild som funktionerar som en knapp. De flesta av Novia Raseborgapplikationens knappelement är dessa bildknappar. Detta för att grafiska elementen från den föregående versionen skulle överföras till den nya versionen av applikationen. ImageButtonelementen deklareras på liknande sätt som ImageViewelementen (se Kod 5). Bilden för knappen skalas med `"android:scaleType='fitXY'"` på detta sätt fyller bilden hela knappens dimensioner oberoende av ursprungsbildens dimensioner.

Kod 5. ImageButtonelement struktur exempel.

```
<ImageButton
      android:layout_width='100dp'
      android:layout_height='50dp'
      android:id='@+id/exempelID'
      android:src='@drawable/bildFilnamn'
      android:scaleType='fitXY'
/>
```

Figur 6 demonsterar hur ett ImageButtonelement kan se ut. Bilden i denna figur är förvrängd eftersom den är kodad att ta upp hela det tillgängliga utrymmet. I denna applikation är alla bildknappar ursprungligen kodade på detta sätt men hela knappen skalas i ett senare skede med Java-kod enligt ursprungsbildens dimensioner så att bilden inte förvrängs.

Figur 6. Demonstration av ImageButtonelementet.

ProgressBarelementet används för att visuellt presentera en funktions framskridande. Elementet kan presentera informationen antingen i en cirkel eller som en balk. Detta element används huvudsakligen för att ange att applikationen laddar eller bearbetar information. Då största delen av ProgressBarelementets funktionalitet redigeras från Java-koden. Därför är det relativt enkelt att ställa upp elementets grund i XML (se Kod 6). Det finns tre olika storlekar av cirkelladdare som definieras med ”progressBarStyleLarge”, ”progressBarStyle” och ”progressBarStyleSmall”. För en horisontell balk används ”progressBarStyle Horizontal”.

Kod 6. ProgressBarelement struktur exempel.

```
<ProgressBar
      android:layout_width='wrap_content'
      android:layout_height='wrap_content'
      android:id='@+id/exempelID'
      android:style'?android:attr/progressBarStyleLarge'
/>
```

Figur 7 demonstrerar alla tre olika storlekar av de runda stilarna för ProgressBarelementet och den horisontella balkstilen. Elementerna är animerade och de olika delarna av ringarna blinkar i varierande mörkhetsgrader. Balken är inte från början animerad men kan programmeras att fyllas i med en anna färg.

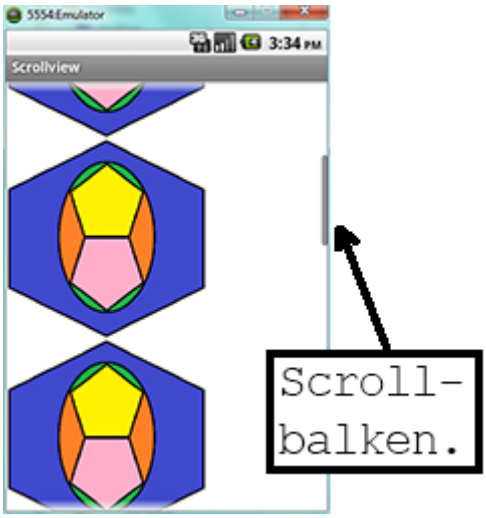
Figur 7. Demonstration av ProgressBarelementet och dess stilar.

ScrollView är ett element som används för att flytta innehåll in och ut ur mobiltelefonens vy. Även om vissa element har en inbyggd skrollnings funktion, kan man med ScrollView elementet låta användaren flytta en samling element som till exempel knappar. ScrollView elementets uppställning är enkelt. I Novia Raseborgapplikationen valdes dock att gömma balkarna som ursprungligen presenteras av Android då användaren skrollar innehållet. Detta görs med `"android:scrollbars='none'"` vilket framgår i Kod 7.

Kod 7. ScrollViewelement struktur exempel.

```
<ScrollView
      android:layout_width='fill_parent'
      android:layout_height='wrap_content'
      android:id='@+id/exempelID'
      android:scrollbars='none'
>
</ScrollView>
```

Figur 8 demonstrerar hur ett ScrollViewelement fungerar. I detta fall har dock inte skrollningsbalkarna gömts. Skrollningsbalken är användbar när man vill visa en stor mängd information så att användarna vet hur långt på sidan de kommit. I flesta fallen används skrollfunktionalitet för att presentera text.



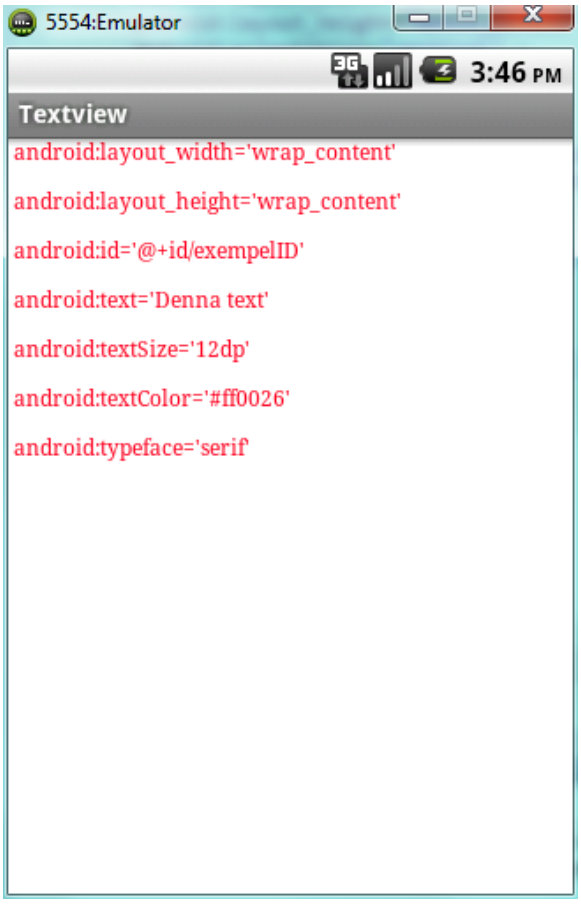
Figur 8. Demonstration av ScrollViewelementet.

TextView är ett element som håller en textsnutt. Texten kan formateras till Android stödda fonter och självdefinierade fonter. Fontstorlek, färg, orientering och dylika parametrar kan enkelt specificeras för TextViewelementet. Exempel på uppställningen av TextViewelementet kan ses i Kod 8.

Kod 8. TextViewelement struktur exempel.

```
<TextView
      android:layout_width='wrap_content'
      android:layout_height='wrap_content'
      android:id='@+id/exempelID'
      android:text='Exempel text.'
      android:textSize='12dp'
      android:textColor='#ff0026
      android:typeface='serif'
/>
```

Figur 9 illustrerar hur TextViewelementet som ställs in i Kod 8 ser ut i praktiken. Exemplet i Figur 9 aviker från Kod 8 med att själva koden tar platsen av ”`'Exempel text.'`” på raden ”`android:text='Exempel text.'`”. TextViewelement används ofta för titlar eller fylls dynamiskt med innehåll genom Java-kod.



Figur 9. Illustration på hur TextViewelementet ser ut.

WebView är ett element som används för att presentera HTML-innehåll. Elementet stöder JavaScript men detta måste skilt specificeras. Det är dock inte rekommenderat att tillåta JavaScript i WebViewelement eftersom detta kan skapa säkerhetsrisker. Eftersom WebViewelementet är bara en behållare för webbdata är XML-koden för elementet mycket enkelt så som kan ses i Kod 9.

Kod 9. WebViewelement struktur exempel.

```
<WebView
      android:layout_width='fill_parent'
      android:layout_height='fill_parent'
      android:id='@+id/exempelID'
      android:orientation='vertical'
/>
```

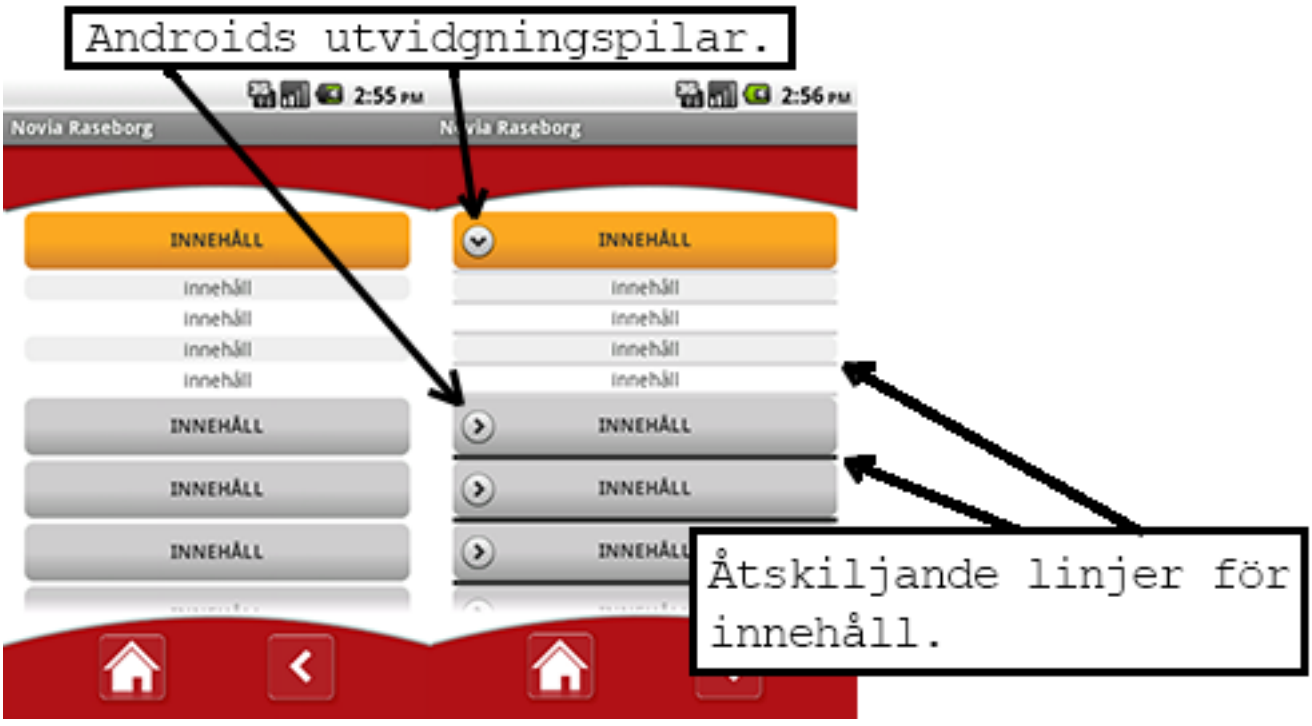
ExpandableListView används för att skapa listor med flere nivåer. Listornas nivåer går att expandera och förminskas så att de underliggande nivåer presenteras och göms i respektive situation. Detta element hjälper användaren att komma åt information snabbt utan att måsta

bläddra igenom oönskad information. ExpandableListViewelementet har några unika parametrar. Kod 10 illustrerar hur dessa parametrar ställs in.

Kod 10. ExpandableListViewelement struktur exempel.

```
<ExpandableListView
      android:layout_width='fill_parent'
      android:layout_height='fill_parent'
      android:id='@+id/exempelID'
      android:scrollbars='none'
      android:groupIndicator='@null'
      android:divider='@null'
      android:dividerHeight='5dp'
/>
```

Med "`android:groupIndicator='@null'`" göms Androids utvidgningspilarna som anger om gruppen är utvidgad eller sluten. Koden "`android:divider='@null'`" gömmer de visuella linjerna som annars visas mellan innehållselementen. Med "`android:dividerHeight='5dp'`" ställs in mellanrummet mellan innehållselementerna. Figur 10 demonstrerar ett ExpandableListViewelement med dessa inställningar och utan dem.



Figur 10. Demonstration av ExpandableListViewelementet.

3.2 Huvudmeny

Huvudmenyn (se Figur 11) är en enkel samling knappar som leder användaren vidare till önskad information eller undermenyer. För att ställa upp knapparna parvis är de inbäddade parvis i LinearLayoutelement. Dessa LinearLayoutelement ordnar innehållet i horisontella riktningen och placerar på så sätt knapparna bredvid varandra. LinearLayoutelementerna är inbäddade i ett ScrollViewelement som tillåter knapparna att skrollas i mobiltelefonens vy när alla knappar inte ryms i rutan samtidigt. ScrollViewelementet ordnar sitt innehåll i vertikala riktningen och därför ordnas knapparna parvis på varandra.

Figur 11. Applikationens huvudmeny.

Menyns knappar är definierade att fylla sin begränsningsruta även om detta förvränger bilden. Knapparna skalas procentuellt, med Javakod, i relation till mobiltelefonens skärmstorlek och originalbildens proportioner så bilden inte förvrängs. Vänstersidans knappar har också en procentuell marginal till vänster för att placera knapparna en aning inåt från vyns sida. De har också högermarginaler för att skilja åt knapparna från högersidans knappar.

Huvudmenyn och alla andra menyer har också en bild med logon för Yrkeshögskolan Novia. Denna logo är endast helt synlig på huvudmenyn. Den flyttas upp ur skärmen så att endast nedre kanten av bilden syns på de andra menyerna. Logon består av ett ImageViewelement som skalas och animeras med Javakod.

Navigationsmenyn finns också på alla menyerna och låter användaren hoppa en sida bakåt och direkt tillbaka till huvudmenyn. Navigationsmenyn består av ett ImageViewelement med två ImageButtonelement placerade på bilden. Knapparna är animerade genom sina egna XML-filer och Java-kod så att de blinkar svarta när användaren klickar dem.

3.3 Infomenyn

Info menyn är en samling knappar som skickar användaren vidare till texter som ger information om skolan, skolans utbildningsprogram och studielivet. Menyn liknar huvudmenyn men knapparna är placerade i en enda stapel istället för parvis.

Info menyns knappar (se Figur 12) är ImageButtonelement inbäddade i LinearLayoutelement på samma sätt som i huvudmenyn. Uppställningen skrollas också på samma sätt som huvudmenyns knappar med hjälp av ett överliggande ScrollViewelement. Till skillnad från huvudmenyn är dock knapparna skilda åt med hjälp av LinearLayoutelementernas toppmarginaler istället för knapparnas egna element.

Figur 12. Info menyn.

Infoknapparnas storlek är beräknat med hjälp av Java-kod för att skapa en dynamiskt sammanhängande layout för olika skärmdimensioner. Först används inbyggda Androidmetoder för att få reda på skärmens bredd (se Kod 11).

Kod 11. Metod i Java som hämtar skärmdimensioner.

```
public static float[] getWindowSize(Activity activity){
	float[] sizes = new float[2];
	Display m = activity.getWindowManager().getDefaultDisplay();
	sizes[0] = (float) m.getWidth();
	sizes[1] = (float) m.getHeight()-
	activity.getWindow().findViewById(Window.ID_ANDROID_CONTENT).getTop
	();
	return sizes;
}
```

För att knapparnas bilder inte blir förvrängda måste ursprungsbildernas dimensioner hämtas vilket framgår i Kod 12.

Kod 12. En metod som hämtar ursprungsbildens dimensioner.

```
public static float[] getButtonImageDimensions(ImageButton button){
	float imageDimensions[] = new float[2];
	imageDimensions[0] = (float)
	button.getDrawable().getIntrinsicWidth();
	imageDimensions[1] = (float)
	button.getDrawable().getIntrinsicHeight();
	return imageDimensions;
}
```

Knappens bredd inställs genom att multiplicera skärmens bredd med 0,66. För att beräkna knappens höjd multipliceras knappens bredd med kvoten av ursprungsbildens höjd dividerat med dess bredd . För att knapparna inte skulle sitta helt fast i varandra lämnas ett mellanrum mellan knapparna som är skärmens bredd multiplicerat med 0,02. På samma sätt multipliceras skärmens bredd med 0,17 för att få knapparna centrerade. Alla dessa inställningar framgår i Kod 13.

Kod 13. Inställning av Info knapparnas dimensioner och placering.

```
private void setButtonSizesAndPlacement(ImageButton button){
        float buttonImageDimensions[] =
        Global.getButtonImageDimensions(button);
        LinearLayout.LayoutParams params = (LinearLayout.LayoutParams)
        button.getLayoutParams();
        params.width = (int)
        Math.round(windowDimensions[0]*buttonWidthpercentage);
        params.height = (int)
        Math.round((windowDimensions[0]*buttonWidthpercentage)*(buttonImage
        Dimensions[1]/buttonImageDimensions[0]));
        params.topMargin = (int)
        Math.round(windowDimensions[0]*buttonSeparationpercentage);
        params.leftMargin = (int)
        Math.round(windowDimensions[0]*buttonEdgeSeparationpercentage);
        button.setLayoutParams(params);
}
```

Innehållet i Info menyn är presenterat med ett WebViewelement som är inbäddat i ett ScrollViewelement för att tillåta skrollandet av texten. Innehållet (se Figur 13) är presenterat i WebViewelementet för att behålla formateringen från servern så långt som möjligt. På detta sätt behålls också textens fontstorlek och dylika parametrar samma som i förra versionen av applikationen. TextViewelementet kunde ha använts för att presentera texten men detta skulle ha begränsat uppdateringsmöjligheterna från serverns sida.

Figur 13. Exempel på innehåll för Info menyn.

3.4 Schema, Kontaker och Webmail menyerna

Schema, Kontakter och Webmail undermenyerna använder samma grafiska uppställning. De har ett WebViewelement som visar HTML-information beroende på vilken knapp som klickats i huvudmenyn. Informationen är en hårdkodad länk i Java och dess innehåll beror på vad uppehållarna av webbsidorna lagat i dem. De är dock relativt säkra länkar då de styr till intranätet för Yrkeshögskolan Novia och deras e-postsystem. När användaren trycker en av knapparna skickas en URL länk till nästa aktivitet med en ”Intent” vilket illustreras i Kod 14.

Kod 14. Knapp funktion med "Intent".

```
@Override
public void onClick(View v) {
	switch(v.getId()){
	case R.id.ibMM3:
	spinner.setVisibility(View.VISIBLE);
	vibrator.vibrate(Global.global_vibrationTime);
	theIntent = new Intent("fi.novia.novia_raseborg.BROWSER");
	theIntent.putExtra("Link_URL",
	"https://intra.novia.fi/kontaktuppgifter/");
	break;
	}
}
```

I en ”Intent” kan man sätta in extra information för olika ändamål. I detta fall används den endast för att bestämma vilken URL som skall laddas. När aktiviteten skapas ställs WebViewelementets innehåll upp med koden i Bilaga 1.

Under betatestfasen slutade länkningen fungera. Det visade sig att problemet var skolans certifikat. Android godkände inte certifikatet vilket resulterade i att ingen information visades på WebViewelementet. För att fixa detta problem måste man förbigå SSL varningen som uppstod när applikationen försökte komma åt webbsidan. Detta gjordes genom att sätta in Kod 15 i klass deklarationen ”`content.setWebViewClient(new WebViewClient(){});`”.

Kod 15. Lösning till certifikat problemet.

```
@Override
public void onReceivedSslError(WebView view, SslErrorHandler bahandlare,
SslError error){
      behandlare.proceed();
}
```

3.5 Vädermenyn

Vädermenyn visar väderinformation samlad vid Yrkeshögskolan Novias Raseborgs Campus. Väderstationen var ett studerandeprojekt för utbildningsprogrammet Automation och IT. Väderstationen är belägen vid Yrkeshögskolan Novias Raseborgs Campus där den samlar väderinformation. Denna information är sedan publicerad på internet på domänen Xively. Denna information nås med applikationen genom att göra en enkel HTML-förfrågan som returnerar JSON-data. Rubriken för denna meny är ett enkelt textelement.

Då värden och deras mängd kan variera har resten av elementen inte definierats i XML-filen för Väder menyn utan dynamiskt med Java. Uppställningen (se Figur 14) består av tre TextView element som är placerade i ett LinearLayout element. LinearLayout elementet har marginaler till vänster och uppåt för att centrera och skilja åt vädervärden från varandra. Det första textfältet för ett värde anger rubriken och var värdet tagits, medan det sista textfältet visar när värdet mätts. Dessa textfält har en liten font storlek och är vänster orienterade. Det mittersta textfältet är större än de andra och är grönfärgad för att det skall vara enkelt att urskilja de mätta värdet.

Figur 14. Vädermenyn.

3.6 Matlistamenyn

Matlista menyns utseende uppdaterades till denna version för att användaren skulle komma åt relevant information snabbt. Matlistans huvudsakliga XML-struktur är enkel. Det är fråga om ett LinearLayoutelement med en instans av ExpandableListViewelementet. ExpandableListViewelementets innehåll deklareras dynamiskt i Java koden.

Då matlistan skall visa knappar (se Figur 15), på den översta nivån av listan, med veckodagarna från måndag till fredag har matlistans layout supplementära XML-filer för knapparnas utseende. Knapparnas utseende definieras i ”drawable” mappen, i hierarkin res/drawable-mdpi/, för både nedtryckta läget som otryckta läget. Andra nivåns listor formateras på samma sätt med varierande bakgrundsfärg så att de är enkla att urskilja.

Figur 15. Matlista menyn.

3.7 Tipsaren

Tipsaren består av två grupper av element inom sina respektive LinearLayoutelement. Första gruppen består av ett TextViewelement som fylls med tips från tips tabellen i databasen (se Kod 16).

Kod 16. Databas förfrågan och fyllning av TextView.

```
ProgressBar spinner = (ProgressBar) findViewById(R.id.pbTipsarenLoader);
spinner.setVisibility(View.GONE);

String data[];
try {
      spinner.setVisibility(View.VISIBLE);
      data = Database.getTIPSARENdata(context);
}catch (Exception e) {
      spinner.setVisibility(View.GONE);
      e.printStackTrace();
}finally {
      spinner.setVisibility(View.GONE);
}

TextView contentView = (TextView) findViewById(R.id.tvTips);
contentView.setText(Html.fromHtml(data[random.nextInt(data.length)]));
```

Texten är inställd att ta upp 75 % av det tillgängliga utrymmet efter att den Navigationsmenyns höjd subtraherats från skärmens höjd (se Bilaga 2). Tipsen ryms i flesta fallen i textfältet men TextViewelementets inbyggda utvidgningsfunktion är aktiverad vid behov för att tillåta användaren att skrolla ifall texten överskrider textfältets storlek.

Andra gruppen är tre ImageButtonelement som ger användaren knappar för att navigera genom tipsen (se Figur 16). Första knappen går tillbaka till förra tipset i databasen, mittersta knappen väljer slumpmässigt ett tips från databasen och den sista knappen går till nästa tips. Denna funktionaliteten förverkligas med onClick metoden vilket framgår från koden i Bilaga 3.

Knapparna är enkla bildknappar vars bild tas från ”drawable” mappen, deras LinearLayout är dock programmerad att ta upp endast 25 % av det tillgängliga utrymmet vilket framgår i Kod 17.

Kod 17. Tipsaren knapparnas storlek och placering.

```
int buttonLayoutHeight = (int) Math.round(contentHeight * 0.25f);
//Placing buttons below content and setting parameters
params = (LinearLayout.LayoutParams) buttonLayout.getLayoutParams();
params.topMargin = (int) (Math.round(Global.global_topBannerDimensions[1]
- (Global.global_topBannerDimensions[1] *
Global.global_topBannerOffsetMultiplier)) + Math.round(contentHeight *
0.75f));
params.height = buttonLayoutHeight;
buttonLayout.setLayoutParams(params);
```

Figur 16 illustrerar hur Tipsaren fungerar ser ut i sin helhet med alla inställningar. Texten är klart i huvudrollen mitt i vyn med knapparna under för att navigera tipsen.

Figur 16. Tipsaren menyn.

3.8 Aktuelltmenyn

Menyn för aktuella nyheter består av ett ListViewelement som skapar en lista med en inledande text (se Figur 17) för alla aktuella nyheter i databasen. Listans objekt formateras skilt med XML-filer från ”drawable” mappen. Alla objekten har rundade rektanglar runt texten med varierande bakgrundsfärg. Listans objekt ställs upp programmatiskt med text från databasen. Då användaren klickar på knapparna skickas de vidare till samma struktur som presenterar HTML-länkarna för Schema, Kontakter och Webmail. Istället för att visa HTML-länkar tas informationen för aktuellt innehållet från databasen.

Figur 17. Aktuellt menyn.

4 Programmering

Alla datorprogram kräver programmering och denna Androidapplikation är inget undantag. Även om Androids programmering tillåter att skapa grova designstrukturer för applikationen med XML-krävs Java-kod för att skapa mera invecklade strukturer och ge applikationen funktionalitet. I denna applikation används Java kod för att skala designen dynamiskt till olika skärmstorlekar, detta för att hålla bättre koll på alla variabler inom Java. Som exempel på detta kan man se Kod 13 som skalar Info menyns knappar.

4.1 Java

Java skapades av Sun Microsystems men utvecklas numera av Oracle eftersom Oracle köpte Sun Microsystems. Java är ett objekt orienterat programmeringsspråk. Ett objekt orienterat språk skapar samlingar av information och funktioner till en enhet. Dessa enheter eller objekt används sedan av programmet för att skapa synliga eller icke synliga objekt som till exempel knappar eller databaser. (Silander, Ollikainen, Peltomäki 2010, s. 128). Java är plattformsoberoende, vilket betyder att då man programmerat ett program fungerar det på alla plattformer som kan installera Java. (Skansholm 2013, s. 5-6).

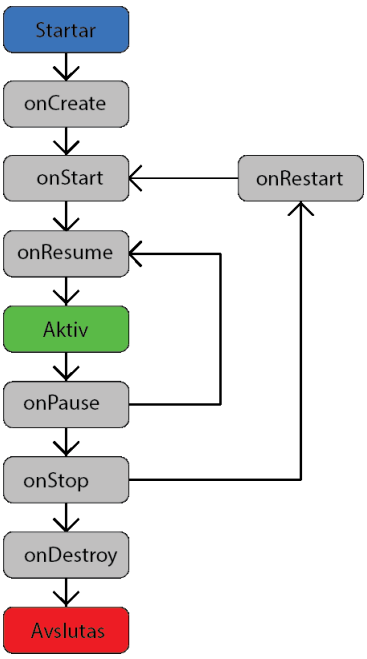
Java har många inbyggda funktioner vilket gör den behändig och snabbt att programmera med. Många väljer att lära sig Java som sitt första programmeringsspråk för att det är enkelt att bygga grafiska användargränssnitt. Java är också behändigare än till exempel

C++ programmeringsspråket som har såkallade ”pekare” för minnesallokering medan Java inte har dessa. (Skansholm 2013, s. 6).

4.2 Android aktivteter och manifestet

Android hanterar sin funktionalitet med aktiviteter. Dessa är delar av programmet som tas fram när de används och flyttas i bakgrunden eller avslutas när användaren går till en annan aktivitet eller stänger av applikationen. (Trangius 2012, s. 20).Om en aktivitet har flyttats till bakgrunden utan att den avslutats kan den återställas. När en aktivitet påbörjas, återställs eller avslutas kan funktionalitet köras. Dessa tillståndsförändringar används ofta för att behandla information som till exempel beräkningar, ladda information från minnet eller ladda ner information från internet.

Då en Android aktivitet startas går den igenom tillståndsmetoderna onCreate, onStart och onResume (se Figur 18). Om aktiviteten avslutas går den igenom tillståndsmetoderna onPause, onStop och onDestroy (se Figur 18). Om aktiviteten avbryts går den igenom onPause-metoden, om aktiviteten fortsätts körs onResume metoden (se Figur 18). Om aktiviteten avbryts och stoppas går den genom både onPause-metoden och onStop-metoden om aktiviteten sedan startas om går den genom onRestart-metoden(se Figur 18). Alla dessa metoder programmeras skilt för varje aktivitet och gör inget från början.



Figur 18. Androidaktiviteternas livscykel.

Största delen av koden som skrivs i Java körs i onCreate metoden för att ställa upp utseendet för aktiviteten då den påbörjas. De andra metoderna används mera sällan men Kod 18 är ett exempel på hur onPause och onResume metoderna används för att återställa en animation.

Kod 18. Aktivitet metoder.

```
//Activity resume settings
@Override
protected void onResume() {
	super.onResume();
	Global.bannerAnimationReset(topBanner);
	enableButtons();
	Global.MainMenuinForground = true;
}

//Activity pause settings
@Override
protected void onPause() {
	super.onPause();
	Global.MainMenuinForground = false;}
```

Androidmanifestet används för information som krävs före applikationen körs. I manifestet kan man till exempel skriva in en beskrivning av applikationen, titel för applikationen och deklarera version numret. I manifestet deklareras aktiviteterna och hur dessa beter sig. Här deklareras till exempel om aktiviteten visas i ”Landscape” eller ”Portrait” vyerna (se Bilaga 4).

4.2 Knappar

Knappar finns i flesta Androidapplikationer. De är enkla att ställa upp för att funktionaliteten finns inbyggd i Android. Det är möjligt att skapa och modifiera knappar i aktiviteten genom Java kod men det är ofta behändigare att skapa knapparnas generella utseende i XML.

För att skapa funktionalitet för knapparna kan man använda onClick metoden. Metoden tar in ett viewelement från vilket man kan få klickade objektets identifikation. Med en enkel

switch metod kan man specifiera vad som händer när en eller flere knappar trycks. Knappar använder sig ofta av Androids Vibrator klass som vibrerar telefonen för att bekräfta att en knapp trycktes före funktionalitet körs. Vanliga funktioner är att bekräfta ett val eller starta en ny aktivitet.

4.3 Navigationsmenyn

Navigationsmenyn består av en balk och två knappar längst nere på Novia Raseborgapplikationen och en balk högst uppe på applikationen. Dessa hänger med i de flesta aktiviteterna i applikationen. Balkarna är huvudsakligen med för att ge mera färg åt applikationen men knapparna är praktiska för att orientera tillbaka till en tidigare aktivitet eller för att återvända tillbaka till huvudmenyn.

Navigationsmenyns element deklareras skilt i varje aktivitets XML-layout och importeras till respektive aktivitets java kod. Funktionaliteten för elementerna körs dock genom en global klass eftersom navigationsmenyns beteende är identisk i nästan varje aktivitet. Huvudmenyn är den enda aktiviteten som har annorlunda beteende för navigationsmenyn då övre balken har en enkel animation som flyttar den uppåt ur vyn.

4.4 Att hämta information via internet

Novia Raseborgapplikationen söker en stor andel av sin information från internet. För att komma åt informationen använder applikationen en såkallad ”Http request” genom att använda elementerna HttpClient, HttpGet, HttpResponse, BufferReader och InputStreamReader. HttpClient skapar en förbindelse till en webbsida. HttpGet skapar en URL-länk till en PHP-fil med verifikations information och ytterligare information för att specificera vilken information som skall skickas tillbaka. HttpResponse kör ingång förbindelsen och tar emot inflödet av information. InputStreamReader elementet tar HttpResponse elementets information som sedan binds till ett enda ”String” element av BufferReader elementet.

När informationen hämtats från servern formateras den om till JSON information vilket är enkelt eftersom informationen är deklarerad i samma format på servern. JSON

informationen är behändig eftersom den fungerar som objekt. Till sist spjälks informationen upp i sina objektdelar och sparas i databasen för senare användning.

4.5 Att spara information i databas

Då applikationen använder mycket information bör denna information sparas i ett strukturerat format. I Novia Raseborgapplikationen sparas informationen i en databas. För att spara information i en databas måste databasen skapas först. Android stöder SQLiteDatabase element.

I Android används en klass som använder SQLiteOpenHelper klassens metoder för att skapa, söka och ta bort information från databasen. För att skapa en tabell körs i onCreate metoden SQL kod så som i
Kod 19. I detta exempel skapas en tabell för att hålla Infomenyns text.

Kod 19. Skapandet av en databas tabell.

```
//Setup INFO table structure
db.execSQL( "CREATE TABLE " + INFO_TABLE +
      " (" +
      INFO_ROW_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
      INFO_ROW_CONTENT + " TEXT NOT NULL, " +
      INFO_ROW_DATA + " TEXT NOT NULL" +
      ");"
);
```

När databasens tabeller ställts upp kan man sätta in information i så som illustreras i Kod 24. I detta exempel går for-loopen igenom hela tabellen för INFO och skriver in ”No data” som ursprungsdata. Detta görs bara för att undvika felaktiga operationer då information hämtas från tabellen för första gången.

Kod 20. Insättning av information i en tabell.

```
//Initial values for INFO table
for(int i = 0; i < INFO_CONTENT.length; i++){
      db.execSQL("INSERT INTO " + INFO_TABLE + " (" + INFO_ROW_CONTENT +
      ", " + INFO_ROW_DATA + ") VALUES('" + INFO_CONTENT[i] + "','No
      data')");
}
```

Då databasens information föråldrats och kräver uppdatering används update metoden och contentValues elementet. Först fylls contentValues elementet med värdena som skall uppdateras. Efter detta uppdateras databasen med det nya värdet så som i Kod 21. I Kod 21 uppdateras tabellen endast om ”INFO_ROW_CONTENT” har samma värde som ”INFO_CONTENT[i]”.

Kod 21. Uppdatering av information i en tabell.

```
ContentValues contentValues = new ContentValues();
for(int i = 0; i < INFO_CONTENT.length; i++){
      contentValues.clear();
      contentValues.put(INFO_ROW_DATA,
      updatedData.getString(INFO_CONTENT[i]));
      applicationDatabase.update(INFO_TABLE, contentValues,
      INFO_ROW_CONTENT + "='" + INFO_CONTENT[i] + "'", null);
}
```

5 Publicering

Även om man kan sprida applikationens fil fritt publiceras flesta Androidapplikationer till Google-play. För att publicera en applikation måste den första kompileras med en keystore-fil. Detta är ett privat certifikat som innehåller information om ägaren av applikationen.

För att skapa filen på Windows börjar man med att öppna filen ”cmd.exe”. När fönstret är öppet bör man navigera till mappen var Java JDK är installerad, vilket är oftast i C:\Program Files\Java\jdkx.x.x\bin\ var ”x.x.x” står för version numret. I denna mapp kan man skriva in ”keytool.exe” för att få mera information. För att skapa certifikatet skriver man in ”keytool –genkey –v –keystore X:\FILNAMN.keystore –alias ALIAS –keyalg RSA –validity DAGAR” där ”X:\FILNAMN” specificerar vart certifikatet skall spara och vad det skall kallas, ALIAS specifierar alias för certifikatet och DAGAR specificerar ett heltal som anger antalet dagar som certifikatet är giltigt. Programmet ber sedan om ett lösenord och bekräftelse på lösenordet. Sedan ber programmet personens förnamn och släktnamn, organisationens enhetens namn, organisationens namn, ort, län, landskoden (två bokstäver) och ber sedan användaren att bekräfta informationen. Sedan ber programmet användaren

skriva in ett lösenord för certifikatet och bekräftelse därefter. Alternativt kan samma lösenord användas för både informationen och certifikatet genom att trycka Enter knappen.

När ett certifikat skapats kan man exportera en APK-fil med Eclipse för Google-play. Detta gör man genom att gå till fliken ”File” sedan ”Export”, utvidgar ”Android” fliken och väljer ”Export Android Application”. Till sist slutför man den ledda publiceringsprocessen.

För att få in applikationen i Google-play måste man logga in med sitt Google-konto till Google Play Developer Console. Här kan man klicka på ”Lägg till en ny app” för att skapa en ny applikation eller granska applikationerna som redan skapats genom att klicka på dem. När man skapar en applikation ber Google-play om viss information som applikationens namn, beskrivning, bilder av applikationen och dylikt. I APK fliken av en applikation kan man också ladda upp nya versioner av applikationen och sätta dem i alpha- och betatestfaser. När en applikation är i en testfas kan man ge en Google-grupp rättigheter att ladda ner och testa applikationen. Detta är nyttigt för att rensa bort fel före publicering.

6 Sammanfattning

Uppdragsgivaren avsikt med Novia Raseborgapplikationen var att skapa ett hjälpmedel för studiehandledning. Applikationens huvudsakliga betydelse var att utrusta studerande med information och idka kommunikation mellan studerande och studiehandledaren på Yrkeshögskolan Novias Campus i Raseborg. I detta projekt skapades en ny Androidversion i Java för att ersätta den begränsade version som skapats med hjälp av PhoneGap. Applikationen förväntades i stort sätt förbli oförändrad i design men förväntades att vara snabbare att använda.

Då applikationen skulle visuellt förbli densamma kunde man använda bildmaterialet från förra versionen på detta sätt behövdes designen ändras endast i Matlista menyn. Matlista menyn ändrades så att innehållet visades som en utvidgbar lista istället för en lång lista. Detta för att presentera informationen i ett behändigare format. Nästan alla texter i applikationen visas i HTML-format för att behålla det gamla utseendet för texterna från förra versionen av applikationen. Undantagen är introduktionssnuttarna för artiklarna i Aktuellt menyn, Tipsarens texter och Matlistans innehåll.

Novia Raseborgapplikationens nya version överfördes att använda Androids inbyggda metoder i den mån som det var möjligt för att optimera applikationen. Till exempel användes en SQLite databas för att spara den extern data som applikationen behöver.

Projektet var framgångsrikt och resulterade i en Androidversion av Novia Raseborgapplikationen utförd i Java. Applikationen var märkbart snabbare än den föregående versionen. Ändringarna som gjordes till designen fungerade lika bra eller bättre än de som fanns i versionen som skapats med PhoneGap.

Källförteckning

Hillo, N., (2014). Novia Raseborg App – ohjauksellinen mobiilisovellus opiskelijoiden älypuhelimiin. i: HAAGA-HELIA ammatikorkeakoulu red. *Arvokas ohjaus*. Vantaa: Multiprint.

Silander, S., Ollikainen, V., Peltomäki, J., (2010). *Java*. Porvoo: WS Bookwell.

Skansholm, J., (2013). *Java direkt med swing*. Lund: Studentlitteratur AB

Trangius, K., (2012). *Programmera appar för Android. En steg för steg-guide för nybörjare*. Lidköping: Thelin Läromedel

Figurföteckning

Kodförteckning

Bilagaförteckning

Bilaga 1: Kod för att ställa upp WebViewelementet med URL.

```
WebView content = (WebView) findViewById(R.id.wvBrowserPage);
//Setting up the web view parameters
content.requestFocus(View.FOCUS_DOWN);
content.getSettings().setJavaScriptEnabled(true);
content.setWebViewClient(new WebViewClient(){

	@Override
	public boolean shouldOverrideUrlLoading(WebView view, String url){
		return false;
	}

	//Setting the loading spinner invisible
	@Override
	public void onPageFinished(WebView view, String url) {
		spinner.setVisibility(View.GONE);
		super.onPageFinished(view, url);
	}

	//Setting the loading spinner visible
	@Override
	public void onPageStarted(WebView view, String url, Bitmap favicon)
	{
		spinner.setVisibility(View.VISIBLE);
		super.onPageStarted(view, url, favicon);
	}
});

//Getting the url from the intent and loading it into the web view
URL = getIntent().getExtras().getString("Link_URL");
content.loadUrl(URL);
```

Bilaga 2: Inställning av dimensionerna för Tipsarens TextViewelement.

```
float windowDimensions[];
windowDimensions = Global.getWindowSize((Activity) this);
float contentHeight = windowDimensions[1] - ((int)
(Global.global_topBannerDimensions[1] *
Global.global_topBannerOffsetMultiplier) + (int)
Global.global_bottomBannerDimensions[1]);
int scrollHeight = (int) Math.round(contentHeight * 0.75f);

//Setting text size
contentView.setTextSize((int)((float) Global.getFontSize((Activity)
context) * 1.5f));

//Calculating maximum amount of lines shown before scroll
maxLines = (int) Math.floor((float) scrollHeight /
contentView.getTextSize());
contentView.setMaxLines(maxLines);

//Enabling scrolling
contentView.setMovementMethod(new ScrollingMovementMethod());

//Sizing and placing the TextView
LinearLayout.LayoutParams params = (LinearLayout.LayoutParams)
contentView.getLayoutParams();
params.height = scrollHeight;
params.topMargin = (int) (Global.global_topBannerDimensions[1] -
(Global.global_topBannerDimensions[1] *
Global.global_topBannerOffsetMultiplier));
params.leftMargin = params.rightMargin = (int) (windowDimensions[0] *
0.05);
contentView.setLayoutParams(params);
```

Bilaga 3: Tipsaren knapparnas funktionalitet.

```
//Button click functions for the activity buttons
@Override
public void onClick(View v) {
	switch(v.getId()){

	//Sets the previous string form the array into the content view
	case R.id.ibTipsarenPrev:
	spinner.setVisibility(View.VISIBLE);
	if(data != null || data.length != 0){
		vibrator.vibrate(Global.global_vibrationTime);
			if(dataPointer > 0){
				dataPointer--;
			}else{
				dataPointer = data.length-1;
			}
			contentView.setText(Html.fromHtml(data[dataPointer]));
		}
		contentView.scrollTo(0, 0);
		spinner.setVisibility(View.GONE);
		break;

	//Sets a random string form the array into the content view
	case R.id.ibTipsarenRand:
		spinner.setVisibility(View.VISIBLE);
		if(data != null || data.length != 0){
		vibrator.vibrate(Global.global_vibrationTime);
		dataPointer = random.nextInt(data.length);
		contentView.setText(Html.fromHtml(data[dataPointer]));
		}
		contentView.scrollTo(0, 0);
		spinner.setVisibility(View.GONE);
		break;

	//Sets the next string form the array into the content view
	case R.id.ibTipsarenNext:
		spinner.setVisibility(View.VISIBLE);
		if(data != null || data.length != 0){
			vibrator.vibrate(Global.global_vibrationTime);
			if(dataPointer < data.length-1){
				dataPointer++;
```

```
                }else{
                        dataPointer = 0;
                }
            contentView.setText(Html.fromHtml(data[dataPointer]));
            }
            contentView.scrollTo(0, 0);
            spinner.setVisibility(View.GONE);
            break;
}
```

Bilaga 4: Exempel på Android Manifest.

```
<manifest xmlns:android=http://schemas.android.com/apk/res/android
package="fi.hemsida.applikationnamn"
      android:versionCode="2"
      android:versionName="1.1.0"
      android:installLocation="auto">
<uses-sdk
            android:minSdkVersion="8"
            android:targetSdkVersion="8" />
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.INTERNET"/>

<application
android:allowBackup="true"
            android:icon="@drawable/ic_launcher"
            android:label="@string/app_name"
            android:theme="@style/AppTheme" >
<activity
            android:name="Splash"
            android:label="@string/app_name"
            android:screenOrientation="portrait"
>
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity
android:label="@string/app_name"
      android:name="MainMenu"
      android:screenOrientation="portrait">
            <intent-filter>
                  <action
android:name="fi.novia.novia_raseborg.HUVUDMENY" />
                   <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
</activity>
</application>
</manifest>
```